# 第2章

# 労働力、在庫と振動

本章では、モデルの開発、分析、そして活用における重要な概念に焦点を当てます。 第１章では、システムダイナミクスによるモデル開発プロセスの概要を学びましたが、第２章では例を挙げてより詳しく開発プロセスを学んでいきます。 第１章と第２章を相互に参照しながら学んでいかれるとより理解を深めることができます。 この章を学んだ後、Vensim でダイナミクスモデルを実際に動かしてみください。 更に理解を深めることができるでしょう。

## 2.1. 背景

第２章で扱う例題に関してその背景について説明します。組み立て式のアルミサッシを生産・販売している会社を想像してみてください。事業全体は非常に順調ですが、生産を増やそうとシフトを組んだ途端、仕事が減って生産能力を遊ばせてしまう、という経験をしばしばしています。通常、このような事態が起こると、市場需要の変化や経済のせいにしがちですが、疑問が残ります。なぜなら過去８年間を遡って調べてみると、**実際には販売量は生産量よりもはるかに安定**しているからです。

あなたの課題は、販売量が安定しているにもかかわらず、増産シフトを組んだ途端に生産能力が余ってしまうのはなぜかを明らかにし、その対策を考えることです。

この問題に取り組むにあたっては、**できるだけ単純化して考えることが重要**です。単純化することには、以下のような多くの利点があります。

- モデルを理解しやすくなる。
- 結果を早く得て、対策の正しさを検証できる。
- 複雑なモデルを完成させてから洞察を得るよりも、単純なモデルから始めて、後から**必要に応じて詳細を追加する**方が効率的である。
- モデリングの初期段階では全体像を把握することが大切ですが、モデルを単純化することで、自然と**全体を概観**できる。

しかし、常に単純なモデルから構築し始めるとは限りません。多くの場合、目の前の問題となっている振る舞いは複雑さの結果であるため、複雑な構造をそのままモデル化しようとすることが常です。確かに Vensim のツールを活用すれば、複雑さに対処できます。とはいえ**問題となっている振る舞いの背景にある本質を掴むまでは、単純性を大切にする**ことを忘れないでください。

開発したモデルが、取り組むべき問題すべてを議論するのに十分でないこともありますが、**そのプロセスを通して問題への理解を深めることは、後々役立つはず**です。これとは反対に、大規模で複雑なモデルに非常に長い時間を費やしたにもかかわらず何も得られなかった、ということもよく起こります。これは、時間の著しい無駄遣いと言えるでしょう。

### 2.1.1. 参照モードで問題の本質を把握する

ここでは、参照モードの概念と参照モードをシステムダイナミクスモデルの開発プロセスプロセスにおける活用方法を理解してください。参照モードは、問題をグラフで表したものです。言葉で表現すれば、「生産は販売ほど安定しない」となるでしょう。これをグラフで描けば図 11 「生産は販売ほど安定しない」の参照モードのようになります。

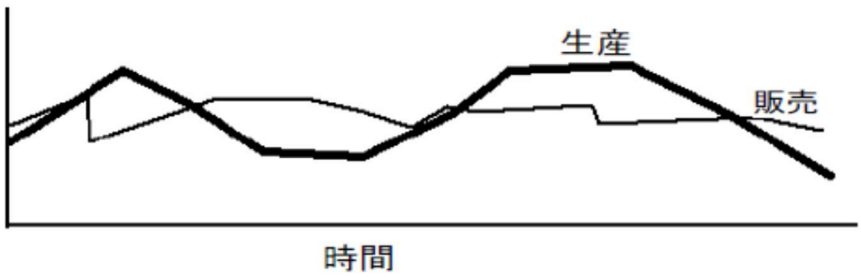


図 *11* 「生産は販売ほど安定しない」の参照モード

この参照モードは、**「これから作成するモデルが示すと考えられる振る舞いをスケッチしたもの」**とも言えます。それは、過去の記録に基づく実際のデータかもしれませんし、新しい状況ではこのようなことが起こるかもしれないという、あなたの想像や期待であるかもしれません。参照モードは、**問題に意識を集中させることで、無駄な思考や試行錯誤を避ける**ために作成します。参照モードを作成したなら、次の目標は参照モードに質的に似ている振る舞いのパターンを生み出すと思われる、**最も単純な構造を見出し定義すること**です。逆に、その構造が適切なものであれば、そのような構造をさらに洗練すれば、利用可能なデータに対して**定量的に検証可能なモデルを開発できる**はずです。

### 2.1.2. 実現性の点検で問題を具体化する

ビジネスがどのように進められるかに関する常識をモデルに組み込みます。実現性の点検における問題の具体化は「常識」レベルで構いません。自分でもまだ十分に把握していない対象を無理に取り扱う必要はありません。先ずは把握できている「常識」を文章化してみてください。例えば、次のようなものです。

- **労働者**なしでは、**生産**はありえない。
- **在庫**なしでは、**出荷**できない。
- **販売**がある期間増加すれば、**生産**を拡大しようとする。
- **在庫**を置かなければ、**売上**を増やせない。

ダイナミック仮説に基づいてシミュレーションモデルを構築する際、これらの実現性の点検に関する情報は常に心に留めておくべき事柄です。積極的に申せば、実現性の点検の条件文に現れる名詞はダイナミクス仮設の中で重要な役割を担う変数である可能性が高いのではないのかという視点で考察してみることは、モデル開発の初期のプロセスにおいて役立つと考えられます。ここでは、労働者、生産、在庫、出荷、販売等です。

### 2.1.3. ダイナミック仮説で構造を考える

ダイナミック仮説を立てるということは「どのような構造であれば、参照モードに示された振る舞いを生み出すことができるのか」を考えることです。この例では、単に参照モードに示されている２つの変数（生産と販売）がどのように関連しているかを考えることにより、ダイナミック仮説を定式化できます。つまり、変数「販売」が与えられたとき、変数「生産」を決定するためのポリシー（あるいはルール）を特定することです。この会社のダイナミック仮説は、マネージャーが現在の販売高に基づいて「生産」量を決めていますが、必要以上に高い（あるいは低い）生産量を増幅して決めてしまっているのではないかというものです。

参照モードには生産と販売という２つの変数が示されています。そして、これら２つの変数をモデルに含めるのが自然でしょう。一般的に参照モードをスケッチすることから始めるのが良いと言われていますが、今回のケースでも生産、販売という変数を中心に考えることになったのは、このことの一つの証左と言えます。

## 2.2. 労働力/在庫モデル(wfinv1.mdl)

変数「生産」と「販売」にはどのような関係があるのでしょうか？ 何かを売る前には、商品を生産する必要があるという意味で密接な関係があります。販売と生産は以下に列挙する 2 つの方法で関連づけられます。

1. **物理的方法**： 生産は販売するための商品を生み出す
2. **情報的方法**： 管理者は売上に基づいて生産に関する判断する

まずは物理的方法からモデル作りを始めましょう。生産された時点では商品はすぐには販売されません。つまり販売されるまで在庫として保管されます。そして販売された時点で商品は在庫から引き落とされることになります。一般的には在庫となるケースだけではありません。需要に対して供給不足の場合は、生産が行われると在庫にはならず直ぐに出荷され、バックログ（受注残）が減少することになりますから、在庫とバックログの組み合わせとなります。このようにして販売と生産が分離されるのです。バックログがモデルの中で使用される場合は、販売の代わりに注文および出荷という概念を考慮するのが良いでしょう。すなわち、在庫とバックログは対となる概念であり、在庫を増減させる取引において在庫を増やすフローを生産、在庫を減らすフローを販売といい、同じくバックログを増減させる取引においてバックログを増やすフローを受注、バックログを減らすフローを出荷というということを整理しておきましょう。

このモデルでは 1 つのストック変数 **Inventory(在庫)** を使用することにします。そして、そのストックの中に入るフローと出るフローを加えます。次に、マージツールを使用して 2 つの変数 **production(生産量)** と **sales(販売量)**をドラッグして、バルブの上にマージします。この様子を下に図 12 ストック変数 Inventory とそこへのインフローとアウトフローとして示します。


図 *12* ストック変数 *Inventory* とそこへのインフローとアウトフロー

ここここで、**Inventory(在庫) をモデルに加えることに決めた「後に」、フローとして既存の変数を付け加えるという手順が大切**なのです。つまり具体的な作業手順はこのようになります。最初に **Inventory(在庫)** というストックを導入することを決めました。そこで、ストックを表す長方形の図形を作成すると同時に **Inventory(在庫)** という名前をつけました。 その後、そのストックに出入りする 2 つのフローを図として書き加えました。

そして、書き加えたフローに、既に定義してあった変数を使って、フローに名前をつけることができたという、一連の手順に注目する価値があります。何か問題を処理するために、まず下絵を描いてからコンピュータに入力するためにモデルとして清書するという手順では、ひらめきを活かしながら問題を処理することはおぼつかないのです。

### 2.2.1. 労働力（Workforce）

次に、**production(生産量)** がどのようにして決定されるかを考える必要があります。長期的な投資を考えるとき、生産能力は非常に重要ですが、生産能力は大変安定しているものです。しかし現実には短い期間に多くの人々が雇われ生産シフトが図られたりします。そのためのこまごまとした事柄を考えるという仕事も付け加わってきます。すなわち、変更時期を決定しなければなりませんし、管理も変わるかもしれませんし、メンテナンスの予定も変わるかもしれません。しかし、まずは近似的に、より多くの人々がいれば、より多くの製品を作ることができると言ってしまうこともできます。スタートとしてここから始めてみるのは良いことです。そこで、ストック **Workforce（労働者数）** をモデルに加えます。実際に **Workforce（労働者数）** を変化させるものには、労働者の雇用、一時帰休、解雇、そして定年退職があります。ここで、単純性が重要であるということを思い出して、これら全てを統合して一つの合成概念 **net hire rate(月当たり雇用数)** にします。**net hire rate(月当たり雇用数)** は、**Workforce（労働者数）** を増加させるだけでなく減少させる場合もあることに注意してください。下に図 13 Inventory に加えて Workforce とそこへの双方向のフロー を示します。

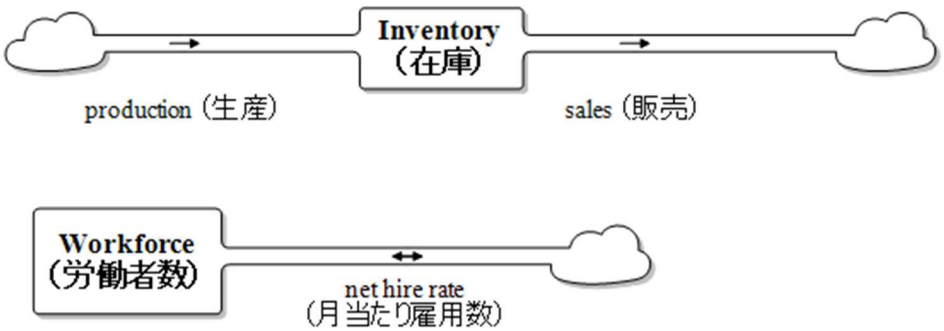


図 *13 Inventory* に加えて *Workforce* とそこへの双方向のフロー

**ワンポイントアドバイス**

**Net hire rate(月当たり雇用数)**は双方向のフローです。このことを明示的にモデル上で表示するためには両矢印 ↔ を用います。まず、ストック **Workforce（労働者数）** から、アウトフローを描きます。次

に、アウトフロー先の雲からストック **Workforce（労働者数）**に向かってアウトフローのパイプに重ねてインフローのパイプを描きます。すると、アウトフローに付されていた→は↔に変化します。

### 2. 2. 2. 振る舞いに関する関係（情報）

2.2.1 項では問題の物理的側面について議論しました。ここからは振る舞いを生じさせる構造のうち情報の連結を定義していきましょう。ある要素が別の要素の振る舞いに影響を与えるという意味で「振る舞いに関する関係づけ」と考えれば良いでしょう。良いモデルを作るためには、物理的ストックとフローのうち重要なものを描くことから始めるのが良いでしょう。このようにしてシステムの中心的な部分を固め、それに基づいてシステムの他の部分を順次概念化していくという手順を踏むことにより、単純化したモデルにできるのです。もう一つのアプローチは、因果ループ図を描き、ストックとフローに書き換えたり、あるいは方程式を直接書き下したりすることです。さらには、因果ループ図を描き、そのうえで再度ストックとフローを描くというやりかたもあります。何が一番良いアプローチであるかは、問題が何であるかということと、分析を行うのが誰であるかということによります。したがって、このガイドでは、章ごとに意図的に異なるアプローチをとるようにします。

情報のつながりを完成させるにあたって、できるだけ物事を単純化する必要があります。生産について考えるにあたって、生産シフトや製造設備に関する話はしまい込んで、**production(生産量)** は **Workforce（労働者数）** に比例するということだけで表現するようにします。そのために、比例定数 **productivity(一人あたり生産性)** を付け加えます。加えて **net hire rate(月当たり雇用数)** は **Workforce（労働者数）** の値に依存して決まるものとします。これらを表すと次の図 14　情報の関係を追加したストック・フロー図　のようになります。

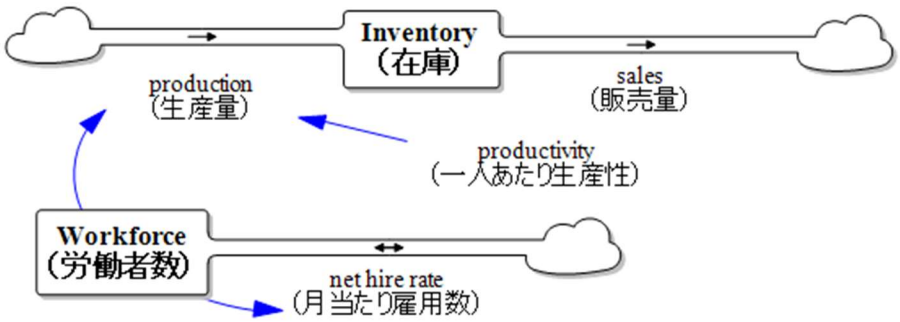


図 *14*　情報の関係を追加したストック・フロー図

**net hire rate(月当たり雇用数)** は雇用される人数を指します。これを定式化する最も直接的な方法が**ストック調整プロセス**です。ストック調整プロセスとは、変数（通常はストック）の現在の値と、目標とする値、あるいはこうあってほしいと希望するレベルを比較し、その差に基づいて行動するものです。例えば、時速 40km で自動車を運転しているとして、時速 50km にしたい場合はアクセルを踏みます。あなたの車の速度（これはストックとみなせます）は、あなたがどれくらいアクセルを踏むか、そして運転している車によって決まるでしょう。

このストック調整プロセスをモデル化するために、変数 **target workforce（目標労働者数）** と **time to adjust workforce(雇用調整期間)**を追加し、それらを組み合わせると次の図 15　ストック調整プロセスを追加したモデル　になります。

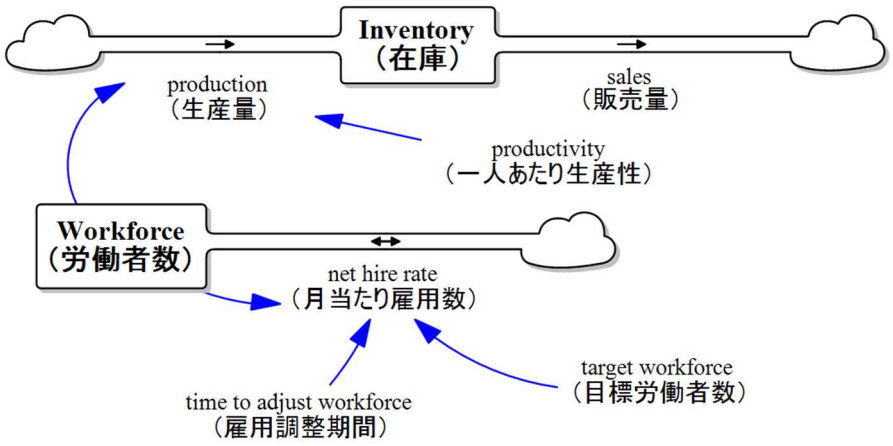


図 *15*　ストック調整プロセスを追加したモデル

**time to adjust workforce(雇用調整期間)** は、マネージャーが労働者数を変更すると決定してから実際に求職者の選考やレイオフの通告を行うまでに要する時間に相当します。**target workforce(目標労働者数)**は、生産に必要となる人員数であり、ストック変数である **Workforce（労働者数）**の初期値として設定されます。次に **target production(目標生産量)** という概念を追加し、それを**target workforce(目標労働者数)**と関連付けます。この**target production(目標生産量)**は **sales(販売量)**に基づいて決定されます。完成した概念モデルを　図 16　完成した概念モデルに示します。

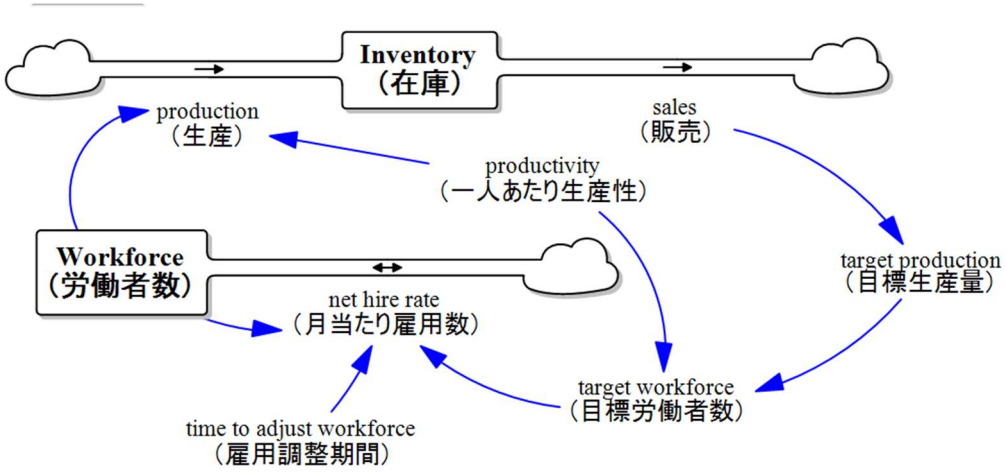


図 *16* 完成した概念モデル

しかしながら、これにはまだ省略した部分がありシミュレーションできません。次のステップは、概念モデルからシミュレーションモデルを開発することです。

### 2.2.3. Vfinv1.mdl の方程式

このモデルの方程式は次のようになります。

**FINAL TIME = 100**
**Units: Month**
**INITIAL TIME = 0**
**Units: Month**
**TIME STEP = 0.25**
**Units: Month**
**SAVEPER = TIME STEP**
**Units: Month**
**Inventory = INTEG(production-sales, 300)**
**Units: Frame**
**net hire rate = (target workforce-Workforce)/time to adjust workforce**
**Units: Person/Month**
**production = Workforce*productivity**
**Units: Frame/Month**
**productivity = 1**
**Units: Frame/Month/Person**

**sales = 100 + STEP (50,20)**
**Units: Frame/Month**
**target production = sales**
**Units: Frame/Month**
**target workforce = target production/productivity**
**Units: Person**
**time to adjust workforce = 3**
**Units: Month**
**Workforce = INTEG(net hire rate, target workforce)**
**Units: Person**

それぞれの方程式は、当初行った概念化の議論と整合しています。**sales(販売)** を定義する式は時間 20 までは 100 で一定であり、その後 150 に段階的に増加し、それ以降も 150 を維持します。この入力パターンは、システムが本当に均衡状態にあるか否かをテストし、新たな運用条件（**sales(販売)** が 150 に増加したこと）に対する調整の様子を確認するために使用します。

このようなステップ入力パターンは、この例で確認しようとしているようなシステム内部で生成されるダイナミクスを観察するための最もシンプルなパターンです。これにより、他に外乱がない場合に、単純な変化がシステム全体にどのように波及していくかを観察できます。

これでモデルは完成です。いよいよシミュレーションに **wfinv1** という名前を付けて実行します。

### 2.2.4. 分析

変数 **production(生産量)**と **Workforce（労働力）**に対する因果グラフは、図 17 production（左） Workforce（右）の因果（原因）グラフのようなダイナミクスを示します。このグラフから、**production(生産量)**や**Workforce（労働者数）**があまり変動せず、100 単位から 150 単位への調整が非常にスムーズに行われていることがすぐに分かります。

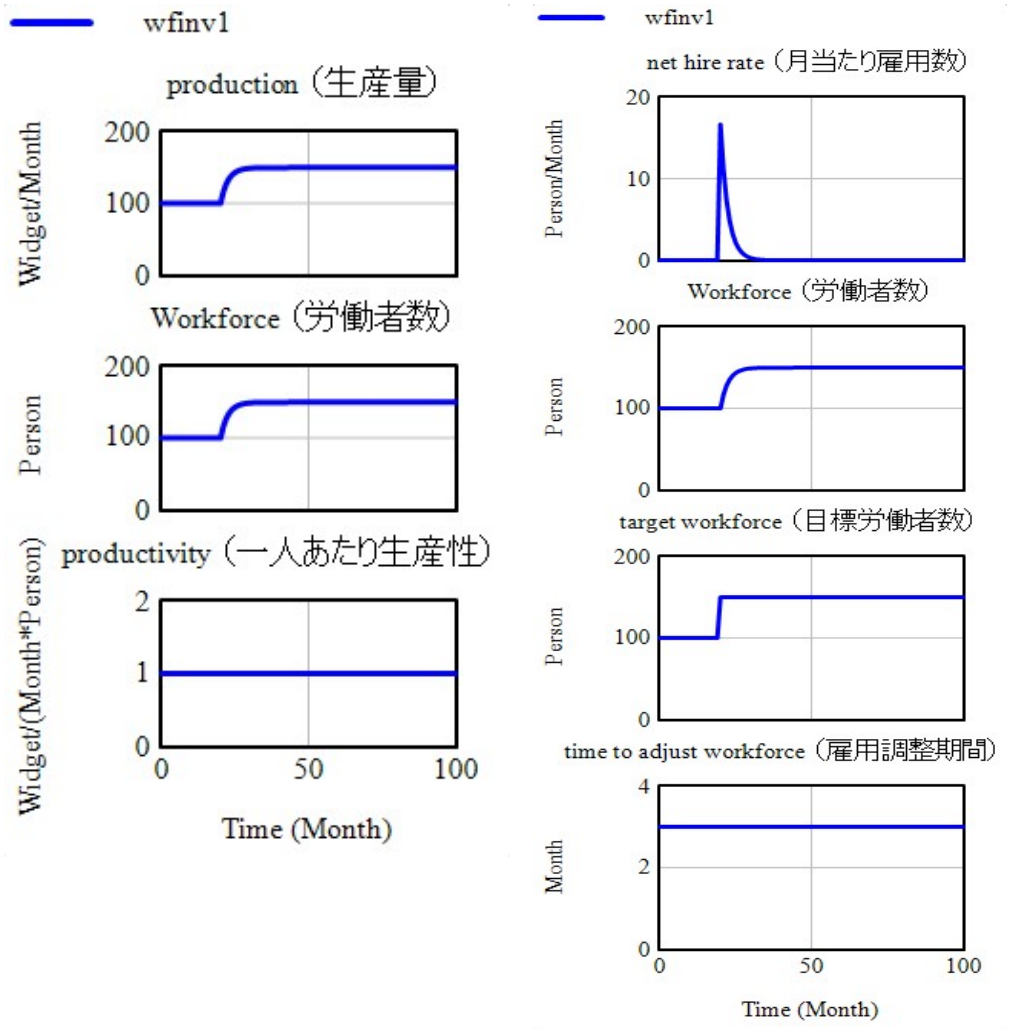


図 *17　production*（左） *Workforce*（右）の因果（原因）グラフ

**ワンポイントアドバイス**

因果グラフを表示するためには、調査対象となる変数をクリックした後、ツールメニューの因果グラフを押下します。 正しく表示されないときは、ツールメニューの因果グラフをクリックしてツールオプションを選択し、因果グラフオプションのメニューの項目のうち、原因／波及の項目の表示が原因変数になっているかチェックください。 なお、図17（左） production(生産量)の原因グラフは、変数**production(生産量)**を調査対象に指定して、原因グラフを押下すれば表示されます。 図１７（右） Workforce（労働者数）の因果グラフは、変数 **Workforce（労働者数）**に代えて **net hire rate（月当たり雇用数）**を指定して因果グラフを作成すると良いでしょう。ストック変数は初期値の設定時を除いて、フローのみがストック変数の変化の原因となりえるので、ストック変数の変化の原因を探る場合は、そのストック変数のフローの変化の原因を探ると良いでしょう。

**Inventory（在庫）**のシミュレーション結果を図 18 Inventory（在庫）のシミュレーション結果に示します。

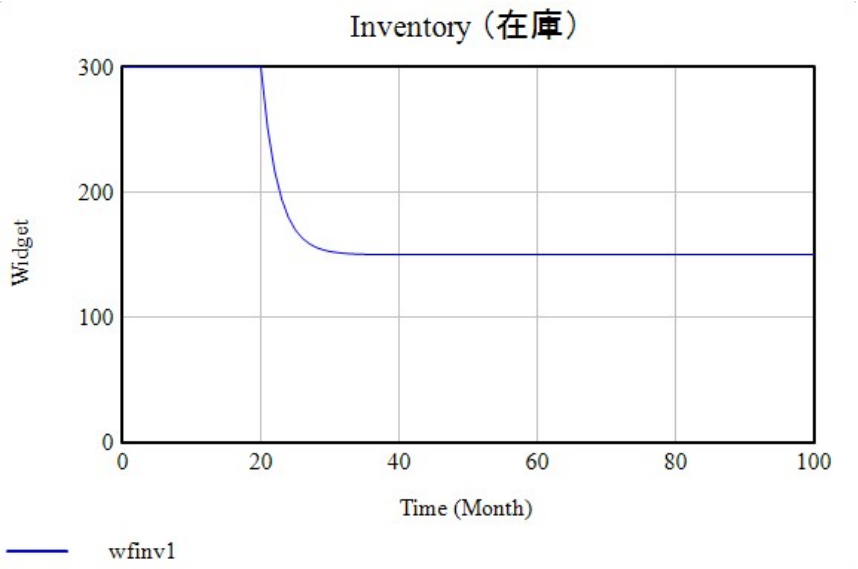


図 *18 Inventory*（在庫）のシミュレーション結果

在庫は初期値 300 から 150 までスムーズに減少します。しかし、在庫を持つ目的が、顧客が製品を迅速かつ適切な構成で入手できるようにすることだとすれば、このような振る舞いは適切ではありません。製品の品揃えを維持し、安心して供給できる在庫水準から乖離している場合は、それを修正する必要がありますが、このモデルではそのような修正は行われていません。

## 2. 3. モデルの改善(wfinv2.mdl)

モデルを改善するために、**target inventory（目標在庫）**と **inventory correction（月当たり在庫調整量）**という 2 つの定数を導入します。考え方はシンプルです。**target inventory(在庫目標)** は販売予測に基づいて決定される目標とする在庫量です。**inventory correction(月当たり在庫調整量)** は、その目標在庫を調整するためのものです。 下に図 19　改善されたモデルとしてこれらの新しいループを図に追加し、太線で強調したものを示します。

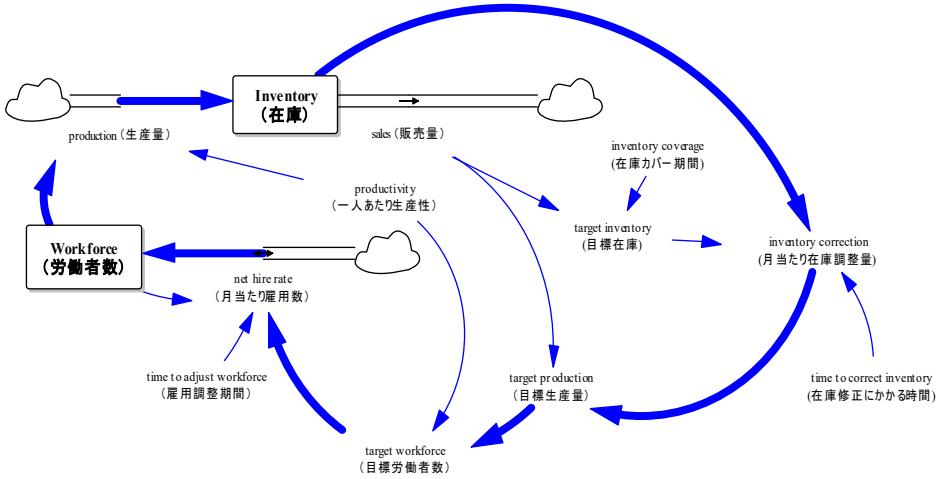


図 *19* 改善されたモデル

### 2. 3. 1. 追加した式

ここで追加した式は下記のとおりです。

**target production = sales + inventory correction**

**Units: Frame/Month**

**inventory correction = (target inventory - Inventory)/TIME TO CORRECT INVENTORY**

**Units: Frame/Month**

**TIME TO CORRECT INVENTORY = 2**

**Units: Month**

**target inventory = sales * INVENTORY COVERAGE**

**Units: Frame**

**INVENTORY COVERAGE = 3**

**Units: Month**

**net hire rate(月当たり雇用数)**と同様に、**inventory correction（月当たり在庫調整量)**もストック調整プロセスを用いて定式化しています。**time to correct inventory（在庫調整期間）**は、著しい在庫変動や生産スケジュールの変更の必要性に気づくまでに要する時間に相当します。

ここで定式化した **inventory correction(月当たり在庫調整量)**と **net hire rate(月当たり雇用数)**の重要な違いは、直接ストックに影響を与えるか否かという点です。**net hire rate(月当たり雇用数)**は **Workforce（労働者数）**を直接調整しようとします。 一方、**inventory correction(月当たり在庫調整量)**は、**target production（目標生産量）**、

**target workforce（目標労働者数）**、**net hire rate(月当たり雇用数)**、**Workforce（労働者数）**、**production(生産量)**といった変数を経由し、最終的に **Inventory（在庫）**にも影響を与えます。この一連の関係の連鎖にはストック変数である **Workforce（労働者数）**が介在しており、それが重要なダイナミクス上の結果をもたらします。

### 2.3.2. 改善後のモデルの振る舞い

モデルを **vfinv2** という名前でシミュレーションします。まず、**Workforce（労働者数）** の振る舞いのグラフを作成してください。その際、データセットとして **wfinv1** と **wfinv2** の両方を読み込みます。その結果を図 20 vfinv2.mdl のシミュレーション結果図 19　改善されたモデルとして示します。


図 *20 vfinv2.mdl* のシミュレーション結果

新しいモデルでは、**Workforce（労働者数）**が安定せず、振動が発生しています。そこで、**Workforce（労働者数）**に対して直接原因トレースを行ってみましょう。さらに**target workforce（目標労働者数）**と**target production（目標生産量）**についても同様に直接原因トレースを行ってみます。結果は図 21 変数target workforceとtarget productionの直接原因トレースの結果のようになります。

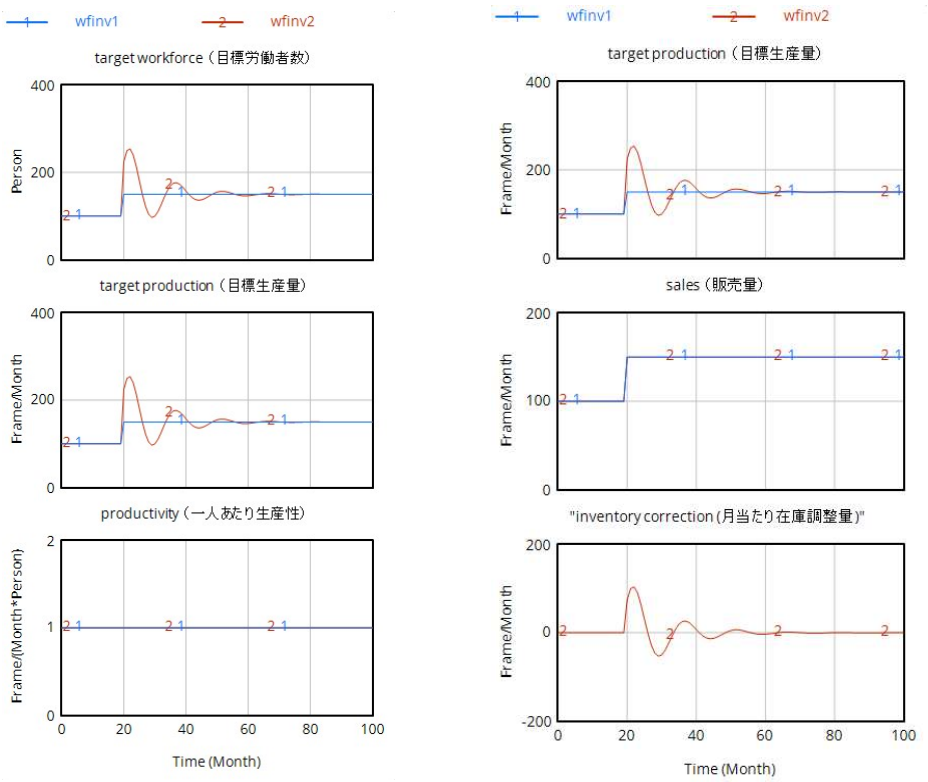


図 *21 変数 target workforce と target production の直接原因トレースの結果*

**Inventory correction(月当たり在庫調整量)**は **wfinv1** モデルには存在しないため、wfinv1 のデータはプロットされていません。**target production(目標生産量)**が振動するのは、在庫水準を修正しようとすることによるものです。**vfinv2** では「在庫」のグラフも同様の振動を示しますが、**wfinv1** とは異なる振る舞いを示します。その様子を図 22 在庫の wfinv1 と wfinv2 のシミュレーション結果の比較に示します。販売が増加したとき、在庫水準は**wfinv1** のように低下するのではなく、新しいより高い望ましい在庫レベルに向けて調整が働きます。

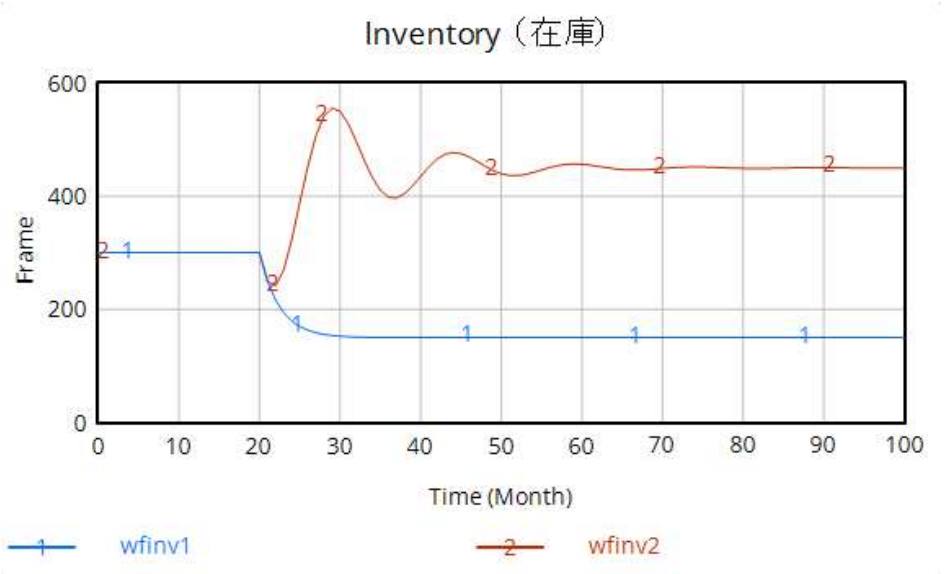


図 *22* 在庫の *wfinv1* と *wfinv2* のシミュレーション結果の比較

### 2.3.3. 位相と振動

このモデルだけでなく振動について一般的に有用な洞察を得るために、制御パネル表示の「自作グラフ」タブを使用して、**Inventory（在庫）**、**target inventory（目標在庫）**、**production(生産量)**、**sales(販売量)**を含むグラフを作成してください。すべての変数のスケールを最小 0、最大 600 に設定します。また、制御パネル表示の「データセット管理」タブで、**wfinv2** を読み込む必要があります。その結果を図 23 位相と振動の考察のためのwfinv2.mdl のシミュレーション結果に示します。

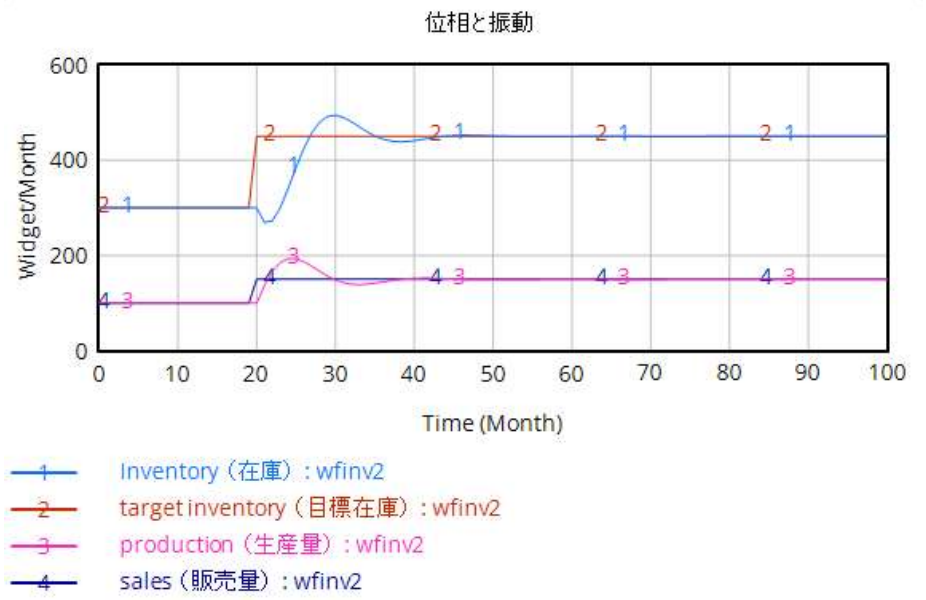


図 *23* 位相と振動の考察のための *wfinv2.mdl* のシミュレーション結果

販売量の急増直後の変化のタイミングに焦点を当てたいので、シフトキーを押しながらグラフの注目したい部分をドラッグするか、コントロールパネルの「時間軸」タブで、表示時間を約 18 ヶ月目から 36 ヶ月目までに設定してください。そうすると、自作グラフは図 24 販売量急増前後の wfinv2.mdl の各変数のシミュレーション結果に示す出力を生成します。

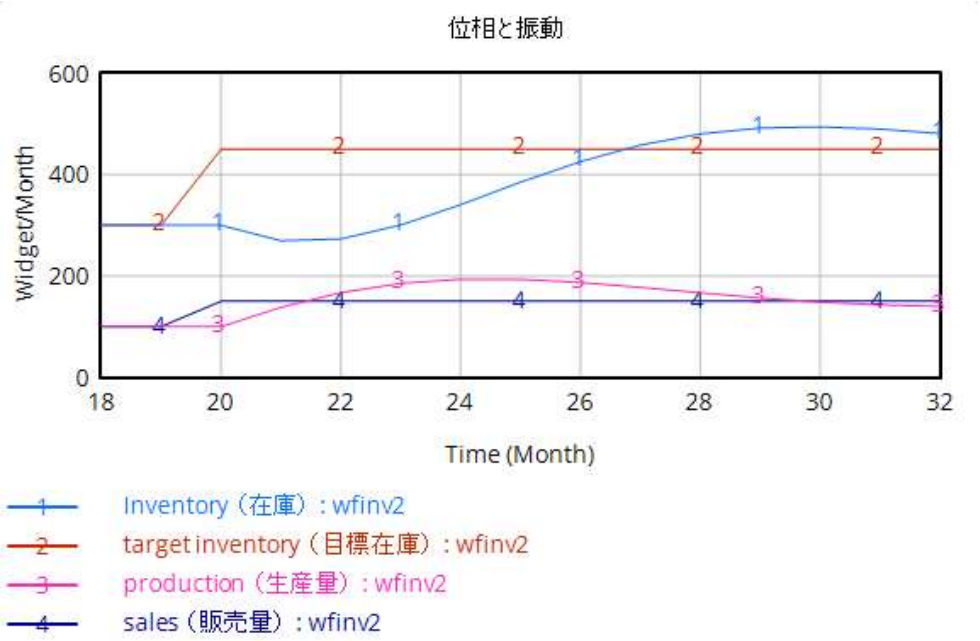


図 *24* 販売量急増前後の *wfinv2.mdl* の各変数のシミュレーション結果

**sales(販売量)**がステップ的に増加すると、**sales(販売量)**が **production（生産量）**を上回るため、**Inventory（在庫）**は減少し始めます。**sales(販売量)**の増加に伴い **target production（目標生産量）**が増加すると、その後 **production(生産量)**も徐々に増加します。これは、**Inventory(在庫)**が減少することで **inventory correction（月当たり在庫調整量）**が増加するためです。

しかし、新規労働者を雇用するのに必要な時間がかかるため、**production(生産量)**を増加させるプロセスには遅れが生じます。その結果、21.3 ヶ月目には **production(生産量)**が **sales(販売量)**と等しくなります。これにより在庫の減少は止まり、徐々に増加に転じます。

**Inventory(在庫)**が増加し、**production(生産量)**も増加します。**Inventory(在庫)**が **target inventory（目標在庫）**を下回っている場合、**production(生産量)**を増加させる圧力がかかります。　しかし、**production(生産量)**が **sales(販売量)**を上回っていても **production(生産量)**は増加し続けます。

**net hire rate (月当たり雇用数)** がマイナスになる前に、現在の在庫の差が **inventory correction(月当たり在庫調整量)** からの圧力を相殺するためには、実際には **production(生産量)** が **sales(販売量)** を十分に上回っている必要があります。**Inventory(在庫)** が **target inventory(在庫目標)** と等しいときでも、**production(生産量)** は **sales(販売量)** よりも多い状態にあることに注意が必要です。なぜなら、労働力に余剰があり、それが引き続き在庫増加の原因となるからです。

**sales(販売量)** よりも **production(生産量)** の変化が大きくなっています。 これは、モデルが、先に参照モードで描いたように、**sales(販売量)** の変化が **production(生産量)** で増幅されることを示しているということになります。

実際のところ販売量の変化が生産に伝わる際、その変化が増幅されることは物理的に避けられません。一時的に生産量の増加は、販売量の増加を上回る必要があります。これは、生産量の調整が行われる前に、(販売量の増加によって) 減少してしまった在庫を回復し、さらに高い目標レベルまで在庫を積み上げる必要があるためです。

**ワンポイントアドバイス**

モデル上で試行錯誤しながら変数の変化を追いかける場合は、ツールメニューから因果グラフやグラフ機能を使うのが便利です。参照したい変数をクリックして指定し、因果グラフやグラフ機能のメニューボタンを押下するだけで、手間と時間をかけずに読みやすいスケールでグラフが表示されます。思考を中断せずに必要な情報を得て、考え、モデルを調整して再試行するという流れを止めることはありません。一方で、上記の例のように、結論的にシステムの挙動を図表としてまとめる場合は、自作グラフを用いると良いでしょう。自作グラフは制御パネルで表示方法を指定する手間と時間がかかりますが、スケールを合わせたり、複数の変数を一つのグラフ上で表示して、それらの相関を表示したりするのに便利です。

### 2.3.4. 変化への対応の感度

パラメータを変更させてモデルで実験することは興味深いものです。会社のパフォーマンスを改善するための施策のひとつは、在庫の変化をより積極的に修正することでしょう。ここで、**time to correct inventory(在庫調整期間)** をより小さな値にしてシミュレートしてみましょう。いま、**time to correct inventory(在庫調整期間)** を 2 ヶ月から 1 ヶ月に減少させる(**wfinv3**)とシミュレーションの出力 **production(生産量)** は次の図 25 time to correct inventory を 1 カ月にした結果に示します。

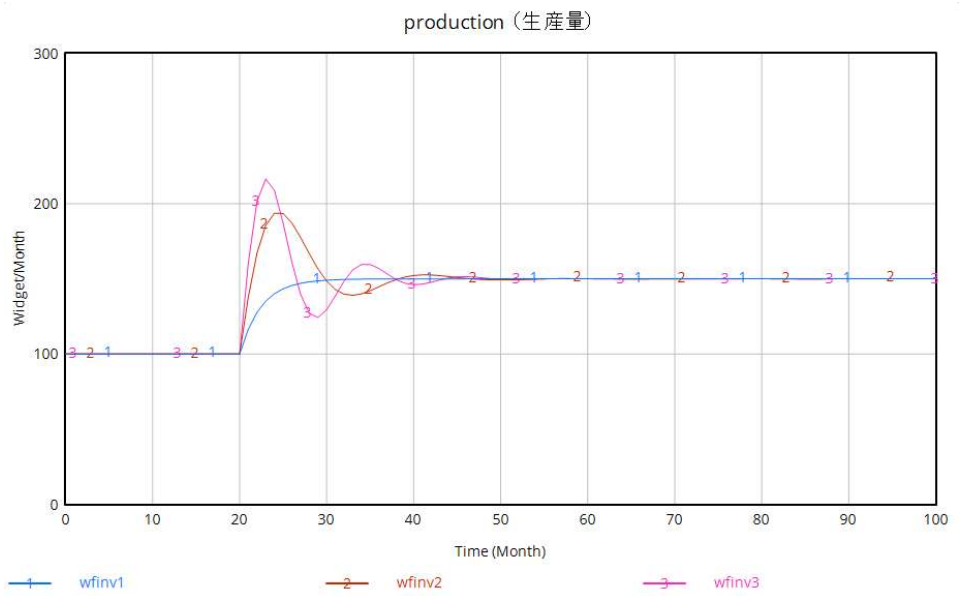


図 *25 time to correct inventory を 1 カ月にした結果*

ここで、在庫を修正する際の遅れを小さくすると、確かに**調整期間は短縮できますが、在庫量の振動の振幅が増加**してしまいます。このような状況は、実際の企業においては望ましくありません。このモデリング演習における、このレスポンスの説明は非常に直観的です。システムは在庫を積み増すために、一時的に目標値を上回る「オーバーシュート」を起こす必要があります。在庫をより速く積み増すには、より大きなオーバーシュート量が必要になります。システム内部に含まれる遅れが原因で、このオーバーシュートは結果的に在庫の過剰な積み増しを引き起こしてしまいます。そして、その過剰な在庫は、今度は生産量を下げる方向に作用します。オーバーシュートが大きければ大きいほど、それに続く生産量の減少もまた大きくなるというわけです。

## 2.4. 発展問題

これまでに作成してきたモデル構造では、実現性の点検項目である「在庫がないと出荷できない」という条件を満たしていないことが分かります。この欠点を修正するため、新たな手を加えてモデル構造を開発してみてください。その際、出荷を明確に表す変数と、出荷の制約を示す非線形関数（通常は表関数が用いられます）を使用すると良いでしょう。

### ワンポイントアドバイス

発展問題への対応を少し考えてみます。現実のビジネスでは、出荷時点で売上が計上されるケースが多いのですが、本章の ***vfinv*** のモデルにおける ***sales(販売)*** は、出荷時点ではなく、現実のビジネスにおける受注の段階で ***sales(販売)*** が計上されています。従って ***sales(販売)*** は ***Inventory(在庫)*** の有無にかか

わらず計上され、その結果***Inventory(在庫)***は負になることもありえます。出題文では出荷の制約を示す非線形関数の利用を示唆されていますが、ここでは処理プロセスと構造をより明確にするため、出荷を表すフロー***shipping(出荷)***を追加するとともに、在庫を割り当てることができなかった***sales(販売)***は受注残として管理し、後日在庫が調達（生産）できた段階で出荷する必要があることから、出荷待ちのストック***waiting(出荷待ち)***を追加します。また、在庫レベルを下げたときの挙動を観察するため、***inventory coverage(在庫カバー期間)***を積極的に調整することを考えて、***Inventory(在庫)***の初期値は固定(***300***)ではなく在庫カバー期間から算出される***target inventory(目標在庫)***とします。

これらを実験するために vfinv2.mdl をもとに修正していくと良いでしょう。変更を加えたモデルを図 26　実現性の検証「在庫がないと出荷できない」を反映したモデル vfinv_advanced.mdl に示します。

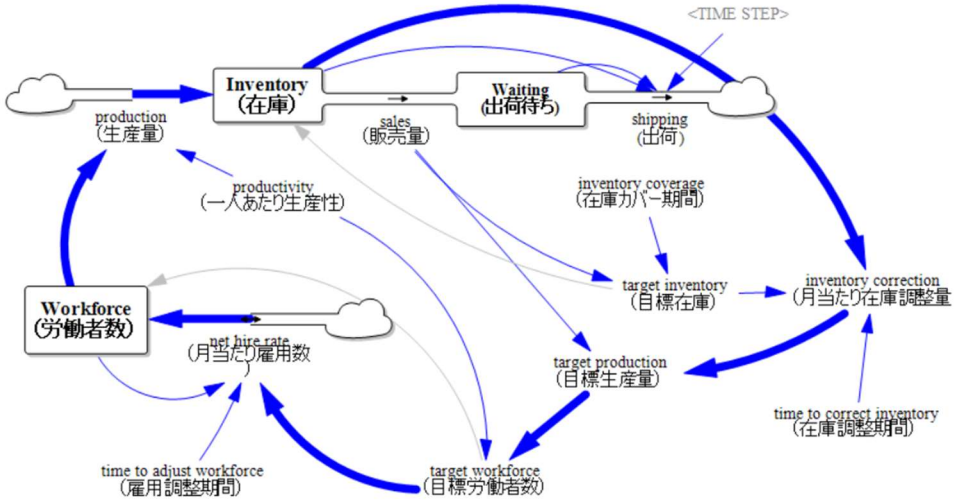


図 *26*　実現性の検証「在庫がないと出荷できない」を反映したモデル *vfinv_advanced.mdl*

追加する式は下記のとおりです。

**shipping (出荷)=IF THEN ELSE( Inventory>=Waiting,**
**MAX (Waiting/TIME STEP ,0),**
**MAX (Inventory/TIME STEP, 0)**
**)**
**UNITS: Widget/Month**
**Waiting= INTEG (sales - shipping, 0)**
**UNITS: Widget**

Inventory(在庫)の初期値を target inventory(目標在庫)とするため下記変更をします。

**Inventory = INTEG (production-sales, target inventory)**
**UNITS: Widget**

このモデルを実行してみましょう。実験のためにはツールメニューにある統合シミュレーションを実行して **inventory coverage(在庫カバー期間)**のスライダーを左右に操作し値を変化させながら、

**Inventory(在庫)**の自作グラフを観察すると良いでしょう。**inventory coverage(在庫カバー期間)**を**0.4** カ月にした結果を図 27 inventory coverage を下げていったときの Inventory と Waiting の挙動に示します。 なお**Inventory(在庫)**は左目盛り、**Waiting(出荷待ち)**は右目盛りを使用します。

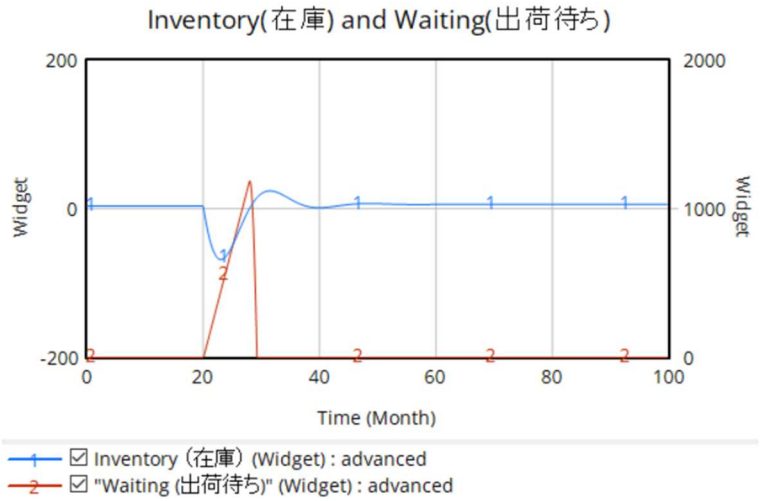


図 *27 inventory coverage* を下げていったときの *Inventory* と *Waiting* の挙動

時刻 20 カ月過ぎに**Inventory(在庫)**が一時的に負となり、その間**Waiting(出荷待ち)**が積みあがっている様子を観察できます。しかし時刻 28 カ月付近で**Inventory(在庫)**が回復すると急速に**Waiting(出荷待ち)**が解消しています。さらに、**inventory coverage(在庫カバー期間)**を下げていき、**0.007** カ月以下になると、**Waiting(出荷待ち)**は解消されなくなり積み上がり続けるようになります。これは業務プロセスが破綻し機能しなくなってしまうことを示唆しています。その様子を図 28 inventory coverage を更に下げていったときの Inventory と Waiting の挙動に示します。なお、**Inventory(在庫)**は左目盛り、**Waiting(出荷待ち)**は右目盛を使用します。

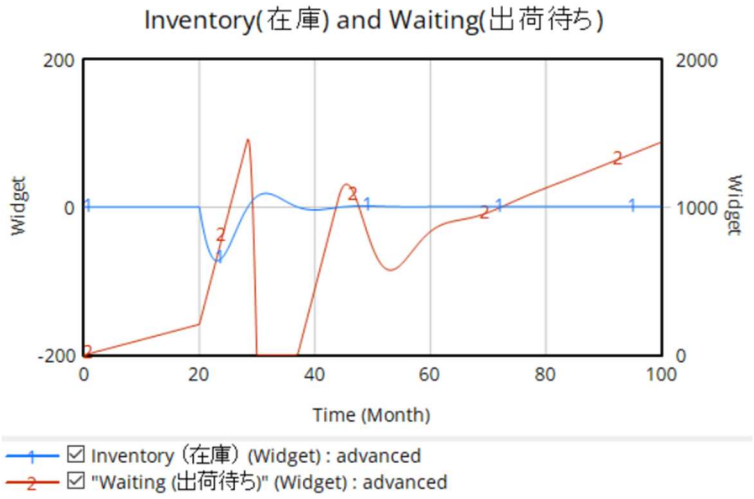


図 *28 inventory coverage* を更に下げていったときの *Inventory* と *Waiting* の挙動

このようにシステムダイナミクスでは、パラメータに極端な値を代入して、モデルの限界を積極的に探ることはとても重要なことです。ここで、**Waiting(出荷待ち)**が積み上がり続けるのは、 **inventory coverage(在庫カバー期間)**を極端に小さい値にすることで、**shipping(出荷)**のスループットが

**sales(販売)**の**150**を常に下回り、**Waiting(出荷待ち)**が積みあがり続けることが原因です。現場において在庫はできるだけ圧縮したいというモチベーションが働きますが、業務の構造をそのままにして**inventory coverage(在庫カバー期間)**を下げようとする努力だけでは、業務が円滑に回らなくなり業務改善は限界に達することが示唆されます。そのような場合は業務構造をそのままにして目標値を上げていくのではなく、業務の構造を抜本的に見直す必要があります。